Définir une grammaire structurelle : DTD

- « DTD » (Document Type Definition)
- La DTD définit la filiation des éléments
  - Quelle est la racine du document?
  - Qui doit/peut avoir quels enfants ?
  - Combien d'enfants ?
  - Qui peut contenir du texte ?
  - Qui a des attributs particuliers ?
  - Quelles sont les valeurs possibles de ces attributs ?
- La DTD est définie dans le document XML par la balise `<!DOCTYPE`
  - interne au document `<!DOCTYPE racine [...]>`
  - dans un fichier externe `<!DOCTYPE racine SYSTEM 'exemple.dtd'>`

Validité d 'un document XML

- Document bien formé (Well Formed document)
  - balises correctement imbriquées
  - parsable et manipulable
  - pas nécessairement valide par rapport à la DTD

- Document valide (Valid document)
  - bien formé + conforme à la DTD (ou au schéma)

DTD : élément et attribut

- <!ELEMENT tag (contenu)
  - Décrit une balise qui fera partie du vocabulaire.
  - ex : <!ELEMENT livre (auteur, editeur)>
- <!ATTLIST tag [attribut type #mode [valeur]]*
  - Définit la liste d'attributs pour une balise
  - ex : <!ATTLIST auteur
    genre CDATA #REQUIRED
    ville CDATA #IMPLIED>
    <!ATTLIST editeur
    ville CDATA #FIXED "Paris">

DTD : Notation

Notations	Exemples
(a, b) séquence	(nom, prenom, rue, ville)
(a|b) liste de choix	(oui|non)
a? élément optionnel [0,1]	(nom, prenom?, rue, ville)
a* élément répétitif [0,N]	(produit*, client)
a+ élément répétitif [1,N]	(produit*, vendeur+)

Types de données

- CDATA
  - Données brutes qui ne seront pas analysées (parsées)
- PCDATA
  - Elément de texte sans descendants ni attributs contenant des caractères à parser
- Enumération
  - Liste de valeurs séparées par |
- ID et IDREF
  - Clé et référence pour les attributs
- ANY
  - Tout texte possible - pour le développement
- EMPTY
  - Vide

Exemple de DTD

```
<!ELEMENT doc (livre* | article+)>
<!ELEMENT livre (titre, auteur+)>
<!ELEMENT article (titre, auteur*)>
<!ELEMENT titre(#PCDATA)>
<!ELEMENT auteur(nom, adresse)>
<!ATTLIST auteur id ID #REQUIRED>
<!ELEMENT nom(prenom?, nomfamille)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT nomfamille (#PCDATA)>
<!ELEMENT adresse ANY>
```

Exemple de DTD interne

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>

<!DOCTYPE ouvrage [
<!-- La DTD est décrite ici -->
<!ELEMENT ouvrage (titre, auteur, chapitre+)>
<!ELEMENT chapitre (titre, section*)>
<!ELEMENT section (titre, para+)>
]>

<ouvrage>
        <titre> Moteurs de recherche </titre>
        <auteur> J. Dupond </auteur>
        <chapitre>
                <titre> accès Web </titre>
                <section >
                        <titre> Introduction </titre>
                        <para> La croissance d'Internet... </para>
                </section>
        </chapitre>
</ouvrage>
```

Exemple de ID et IDREF

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DOCUMENT [
<!ELEMENT DOCUMENT(PERSONNE*)>
<!ELEMENT PERSONNE (#PCDATA)>
<!ATTLIST PERSONNE PNUM ID #REQUIRED>
<!ATTLIST PERSONNE MERE IDREF #IMPLIED>
<!ATTLIST PERSONNE PERE  IDREF #IMPLIED>
]>
<DOCUMENT>
    <PERSONNE PNUM = "P1">Marie</PERSONNE>
    <PERSONNE PNUM = "P2">Jean</PERSONNE>
    <PERSONNE PNUM = "P3" MERE="P1" PERE="P2">Pierre</PERSONNE>
    <PERSONNE PNUM = "P4" MERE="P1" PERE="P2">Julie</PERSONNE>
</DOCUMENT>
```

DTD externe

- Modèle pour plusieurs documents
  - partage des balises et structures

- Définition locale ou externe
  - <!DOCTYPE doc SYSTEM "doc.dtd">
  - <!DOCTYPE doc PUBLIC
    "www.e-xmlmedia.com/doc.dtd">

- Exemple de document

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE VINS SYSTEM "vins.dtd">
...
```

Insuffisance des DTD

- Pas de types de données
  - difficile à interpréter (coercition nécessaire)
  - difficile à traduire en schéma objets
- Pas en XML
  - langage spécifique
- Propositions de compléments
  - XML-data de Microsoft (BizTalk)
  - XML-schema du W3C

Le typage XML avec 'XML schéma'

XML Schéma

- Un schéma d'un document définit:
  - les éléments possibles dans le document
  - les attributs associés à ces éléments
  - la structure du document et les types de données
- Le schéma est spécifié en XML
  - pas de nouveau langage
  - balisage de déclaration
  - domaine spécifique xsd:
- Présente de nombreux avantages
  - structures de données avec types de données
  - extensibilité par héritage et ouverture
  - analysable par un parseur XML standard

Objectifs des schémas

- Reprendre les acquis des DTD
  - Plus riche et complet que les DTD
- Permettre de typer les données
  - Eléments simples et complexes
  - Attributs simples
- Permettre de définir des contraintes
  - Existence, obligatoire, optionnel
  - Domaines, cardinalités, références
  - Patterns, ...
- S'intégrer à la galaxie XML

Principes généraux des schémas

- **Un schéma XML est un document XML.**
- <?xml version="1.0" encoding="ISO-8859-1"?>
- <xsd:schema
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <!-- Déclaration de deux types d 'éléments -->
- <xsd:element name="nom" type="xsd:string" />
- <xsd:element name= "prenom" type="xsd:string" />
- </xsd:schema>

Exemple d'adresse postale en XML: le document

```
<?xml version="1.0"?>
<Addresse_postale_France pays="France">
   <nom>Mr Jean Dupont</nom>
   <rue>rue Camille Desmoulins</rue>
   <ville>Paris</ville>
   <departement>Seine</departement>
   <code_postal>75600</code_postal>
</Addresse_postale_france >
```

DTD d'une adresse postale

```
<!DOCTYPE une_DTD_adresse
[   <!ELEMENT Addresse_postale_france
      (nom, rue, ville, département, code_postal)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT rue (#PCDATA)>
<!ELEMENT ville (#PCDATA)>
<!ELEMENT département (#PCDATA)>
<!ELEMENT code_postal (#PCDATA)>
<!ATTLIST Addresse_postale_france
      pays  NMTOKEN #FIXED 'France' > ]>
```

Typage d'une adresse postale au moyen d'un schéma XML

```
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:complexType name="Addresse_postale_france" >
        <xsd:sequence>
            <xsd:element name="nom" type="xsd:string" />
            <xsd:element name="rue" type="xsd:string" />
            <xsd:element name="ville" type="xsd:string" />
            <xsd:element name="departement" type="xsd:string" />
            <xsd:element name="code_postal" type="xsd:decimal" />
        </xsd:sequence>
         <xsd:attribute name= "pays" type="xsd:NMTOKEN"
                use="fixed" value="FR"/>
</xsd:complexType>      </xsd:schema>
```

Le langage XML schémas : Les composants primaires

- **Un schéma XML est construit par assemblage de différents composants.**

- **Composants de définition de types**
  - Définition de types **simples (Simple type).**
  - Définition de types **complexes (Complex type).**
- **Composants de déclaration**
  - **Déclaration d'éléments.**
  - **Déclaration d'attributs.**

Déclaration des éléments

- **Un élément XML est déclaré par la balise 'element'** de XML schéma qui a de nombreux attributs.
- Les deux principaux attributs sont:

  - **name** : Le nom de l'élément (de la balise associée).

  - **type** : Le type qui peut être simple ou complexe.

Exemple de base

```
<xsd:element name="code_postal" type="xsd:decimal"/>
```

Déclaration des attributs

- **Un attribut est une valeur nommée et typée associée à un élément.**
- **Le type d'un attribut défini en XML schéma est obligatoirement simple.**

```
<xsd:complexType name="TypeRapport">
  <xsd:attribute name= "Date_creation"
   type="xsd:date"/>
    ……
</xsd:complexType>
<xsd:element name="Rapport" type="TypeRapport"/>
```

Autres attributs

- **L'élément attribute de XML Schema peut avoir deux attributs optionnels** : **use** et **value**.
- **On peut ainsi définir des contraintes** de présence et de valeur.
- Selon ces deux attributs, la valeur peut:

  être obligatoire ou non

  être définie ou non par défaut.
- Exemple: <xsd:attribute name= "Date_peremption" type="xsd:date" use="default" value= "2005-12-31"/>

Valeurs possibles pour use

- **Use = required** : L'attribut doit apparaître et prendre la valeur fixée si elle est définie.
- **Use= prohibited** : L'attribut ne doit pas apparaître.
- **Use = optional** : L'attribut peut apparaître et prendre une valeur quelconque.
- **Use= default** : Si l'attribut à une valeur définie il la prend sinon il prend la valeur par défaut.
- **Use= fixed** : La valeur de l'attribut est obligatoirement la valeur définie.
- **Exemple** : <xsd:attribute name= "Date_creation" type="xsd:date" use="required"/>

Types simples

- **'SimpleType' permet de définir des éléments ou des attributs non structurés** : dérivés d'une chaîne, d'un entier etc
  - **Types simples prédéfinis** au sens de la norme XML Schémas ' datatypes': string, integer, boolean ...

```
<xsd:element name="code_postal "
type="xsd:integer"/>
```

  - **Types simples définis par dérivation d'un autre type simple**, au moyen de l'élément <xsd:simpleType ...>

Types complexes

- **Déclarés au moyen de l'élément** <xsd:complexType name="...
- **Ils peuvent contenir d'autres éléments**, des attributs.
- **Trois façons de composer des éléments** dans un type complexe: sequence, choice, all.

Types complexes: Sequence

- Un type **sequence** est défini par une suite de sous-éléments qui doivent être présents dans l'ordre donné.
- Le nombre d'**occurences** de chaque sous-élément est défini par les attributs minOccurs et maxOccurs.

```
<xsd:complexType name= "Commande">
    <xsd:sequence>
        <xsd:element name= "Ad_livraison" type="Addresse"/>
        <xsd:element name= "Ad_facturation" type="Addresse"/>
       <xsd:element name=  "texte" type="xsd:string" minOccurs="1" />
        <xsd:element name="items" type="Items" maxOccurs= "30" />
    </xsd:sequence>
</xsd:complexType>
```

Types complexes: Choice

- Un seul des éléments listés doit être présent.
- Le nombre d'occurences possible est déterminé par les attributs minOccurs et maxOccurs de l'élément.

```
<xsd:complexType name=  "type_temps">
  <xsd:choice >
   <xsd:element name= "Noire"  type="Note" minOccurs="1"
   maxOccurs="1" />
   <xsd:element name= "Croche"  type="Note" minOccurs="2"
   maxOccurs="2" />
  </xsd:choice>
</xsd:complexType>
```

Types complexes: All

- C'est une composition de type ensembliste. Dans un document conforme, les éléments listés doivent être tous présents au plus une fois. Il peuvent apparaître dans n'importe quel ordre.

```
<xsd:complexType name= "Commande">
   <xsd:all>
      <xsd:element name= "Ad_livraison" type="Addresse"/>
      <xsd:element name= "Ad_facturation" type="Addresse"/>
      <xsd:element name= "texte" type="xsd:string" minOccurs="0" />
      <xsd:element name="items"  type="Items" maxOccurs= "30" />
   </xsd:all>
</xsd:complexType>
```

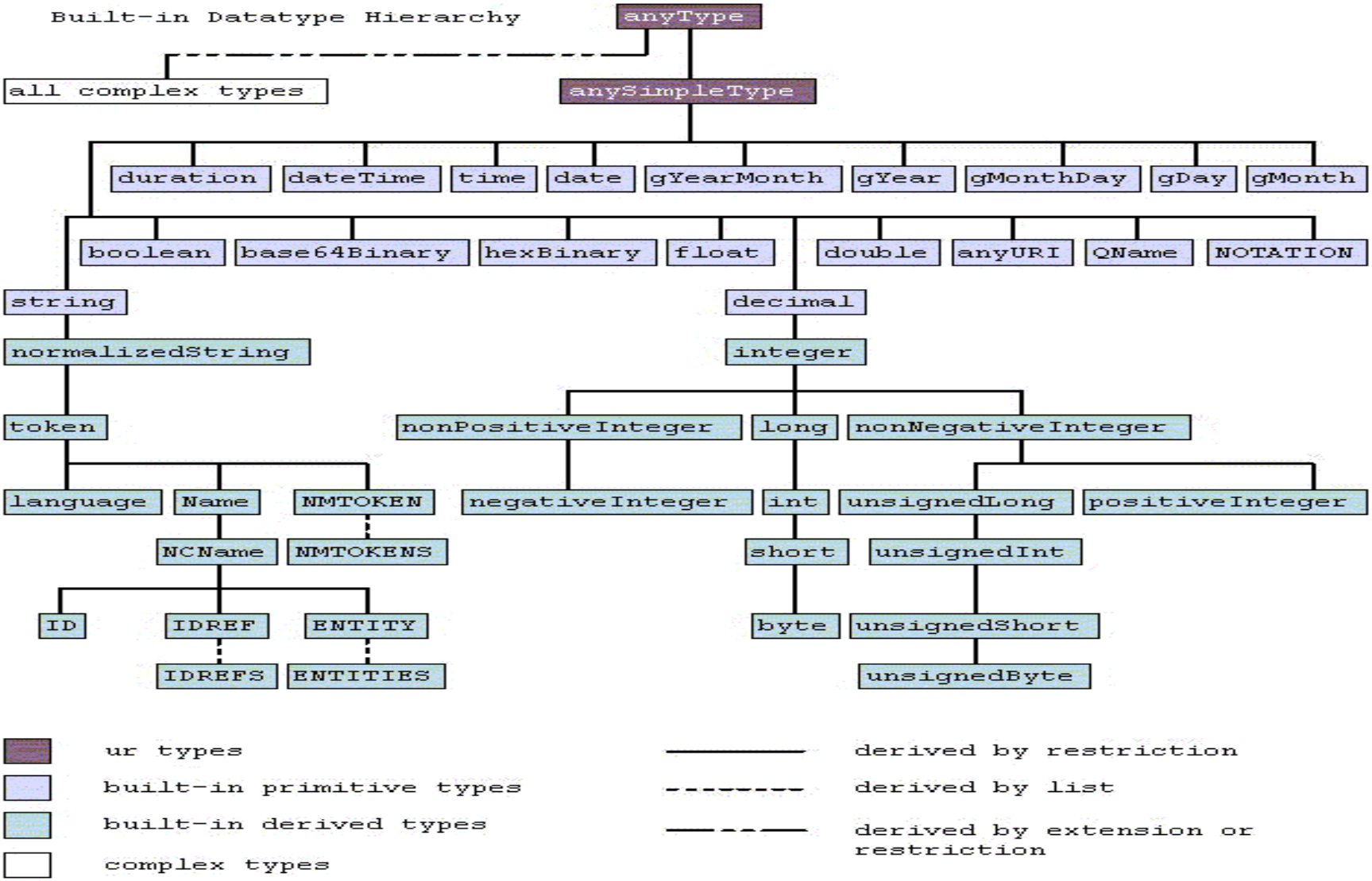
Types: Objectifs de la définition des types

- **Fournir des types primitifs** analogues à ceux qui existent en SQL ou en Java.
- **Définir un système de typage suffisamment riche** pour importer/exporter des données d'une base de données.
- **Distinguer les aspects reliés à la représentation lexicale des données** de ceux gouvernant les ensembles de données sous-jacents.
- **Permettre de créer des types de données usagers** dérivés de types existants en contraignant certaines propriétés (domaine, précision, longueur, format).

Définitions relatives aux types

- **Types primitifs** ('Primitive') Non défini en référence à d'autres types..
- **Types dérivés** ('Derived') Définis par dérivation à partir d'autres types.
- **Types prédéfinis** ('Built-in') Définis dans le cadre de la spécification XML Schéma datatypes (primitif ou dérivé).
- **Types usagers** ('User-derived') Types construits par les utilisateurs.
- **Types atomiques** ('Atomic') Types indivisibles du point de vue de la spécification XML schéma.
- **Types listes** ('List') Types dont les valeurs sont des listes de valeurs de types atomiques.
- **Types unions** ('Union') Types dont les ensembles de valeur sont la réunion d'ensemble de valeurs d'autres types.

Hiérarchie des types prédéfinis



Quelques types prédéfinis

Type	Forme lexicale
String	**Bonjour**
boolean	{**true**, **false**, **1**, **0**}
float	**2345E3**
double	**23.456789E3**
decimal	**808.1**
dateTime	**1999-05-31T13:20:00-05:00.**
binary	**0100**
uriReference	**http://www.cnam.fr**
TOKEN	**un token selon definition des DTD**
….	

Dérivation de types simples
1- Dérivation par restriction

- **La dérivation par restriction restreint l'ensemble des valeurs** d'un type pré existant.
- La restriction est définie par des contraintes de facettes du type de base: valeur min, valeur max ...
- **Exemple :**

```
<xsd:simpleType name= "ChiffresOctaux">
 <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value= 7" />
 </xsd:restriction>
</xsd:simpleType>
```

Les contraintes de facettes

- **lenght** : la longueur d'une donnée.
- **minLenght**: la longueur minimum.
- **maxLenght**: la longueur maximum.
- **enumeration**: un ensemble discret de valeurs.
- **maxInclusive:** une valeur max comprise.
- **maxExclusive**: une valeur max exclue.
- **minInclusive**: une valeur min comprise.
- **maxInclusive**: une valeur min exclue.
- **totalDigits**: le nombre total de chiffres.
- **fractionDigits**: le nombre de chiffres dans la partie fractionnaire.

Exemple d'une énumération

```
<xsd:simpleType name= "Mois">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value= "Janvier"/>
                <xsd:enumeration value="Février"/>
                <xsd:enumeration value="Mars"/>
                <!-- ... -->
        </xsd:restriction>
</xsd:simpleType>
```

2 - Dérivation par extension

- Dériver un nouveau type par **extension** consiste à ajouter à un type existant des sous-éléments ou des attributs.
- On obtient inévitablement **un type complexe**.
- **Exemple**

```
<xsd:complexType name= "mesure">
  <xsd:simpleContent><xsd:extension base="xsd:Decimal">
      <xsd:attribute name="unite" type="xsd:NMTOKEN"/>
  </xsd:extension></xsd:simpleContent>
</xsd:complexType>
 <xsd:element name= "temperature" type= "mesure"/>
 <temperature unit="Kelvin">230</temperature>
```

3 - Dérivation par union

- Pour créer un nouveau type on effectue l'union ensembliste de toutes les valeurs possibles de différents types existants.
- Exemple:

```
<xsd:simpleType name="TransportFormatCaracteres">
<xsd:union memberTypes="base64Binary hexBinary"/>
</xsd:simpleType>
```

4 - Dérivation par liste

- Une **liste** permet de définir un nouveau type de sorte qu'une valeur du nouveau type est une liste de valeurs du type pré existant (valeurs séparées par espace).
- **Exemple**

```
<simpleType name= 'DebitsPossibles'>
        <list itemType='nonNegativeInteger'/>
</simpleType>
<debitsmodemV90 xsd:type='DebitsPossibles'>
33600 56000 </debitsmodemV90>
```

Conclusion typage avec XML schéma

Un standard très utile en réseaux/systèmes répartis

- **Indispensable comme outil d'interopérabilité en univers réparti.**
  - Entre des applications WEB.
  - Dans des approches objets répartis comme SOAP ('Simple Object Protocol')
  - WSDL ('Web Service Definition Language').
  - Entre des bases de données hétérogènes.
- **Quelques reproches**
  - Les performances.
  - Les imperfections à découvrir dans les choix de conception du système de typage.

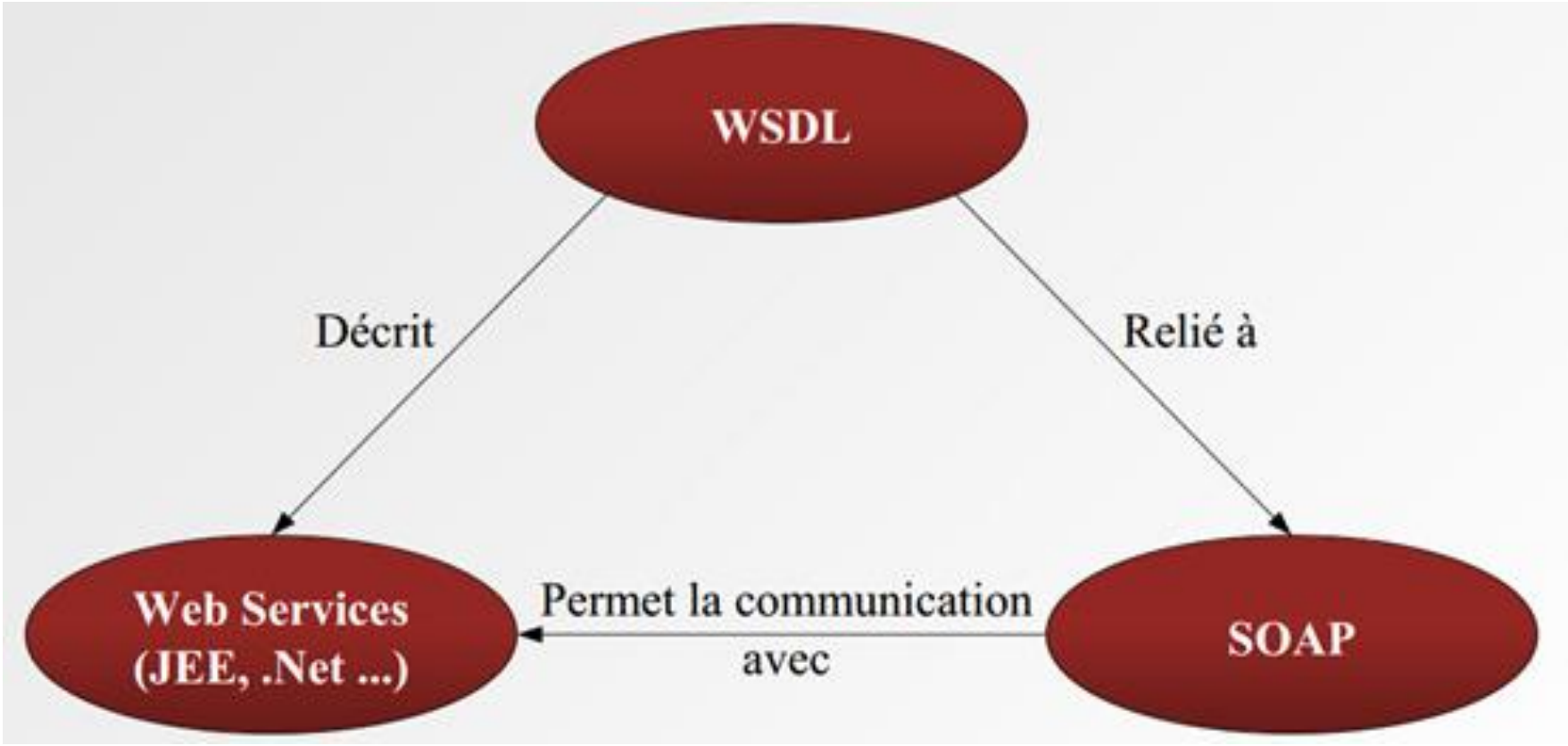
Domaines d'utilisation

- Publication d'informations sur le WEB.
- Commerce électronique.
- Gestion de documents traditionnels.
- Assistance à la formulation et à l'optimisation des requêtes en bases de données.
- Transfert de données entre applications en réseaux.
- Contrôle de supervision et acquisition de données.
- Échange d'informations de niveau méta.

WSDL

- WSDL (Web Service Definition Langage) est un format de représentation des interfaces de services Web en XML.
- WSDL est la représentation XML du langage IDL (description d'interfaces).
- WSDL a deux rôles prépondérants:
  - Il sert de référence à la génération de Proxies.
  - Il assure le couplage entre le client et le serveur par le biais des interfaces.

WSDL


WSDL
Décrit
Relié à
Web Services
(JEE, .Net ...)
Permet la communication
avec
SOAP

Structure d'un document WSDL

Définitions abstraite (Interface):

- <Type>: fournit la définition de types de données utilisés pour décrire les messages échangés
- <message>: représente une définition abstraire (noms et types) des données en cours de transmission.
- <operation>: c'est la description d'une action exposée dans le port.
- <portType>: décrit un ensemble d'opérations. Chaque opération a zéro ou un message en entrée, zéro ou plusieurs messages de sortie ou d'erreurs.

WSDL

Définition concrète (implémentation)

- <binding> : définit le protocole à utiliser pour invoquer le service web
- <port> : spécifie l'emplacement actuel du service
- <service> : décrit un ensemble de points finaux du réseau

Aussi

- <import> utilisé pour faire référence à d'autre documents XML

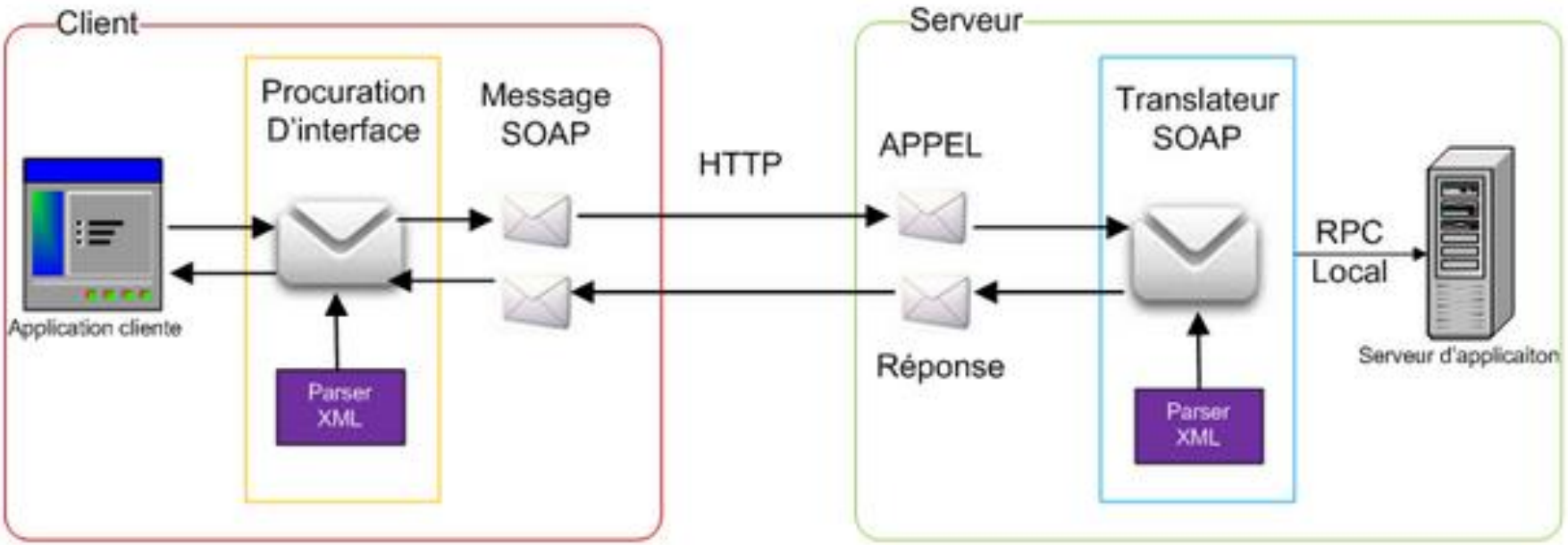
SOAP

- SOAP (Simple Object Access Protocol) est un protocole d'invocation de méthodes sur des services distants.
- Basé sur XML, SOAP est un format de communication pour assurer communication de machine à machine.
- Le protocole permet d'appeler une méthode RPC (Remote Procedure Call) et d'envoyer des messages aux machines distantes via HTTP.
- Ce protocole est très bien adapté à l'utilisation des services web car il permet de fournir au client une grande quantité d'informations récupérées sur un réseau de serveurs tiers.

SOAP

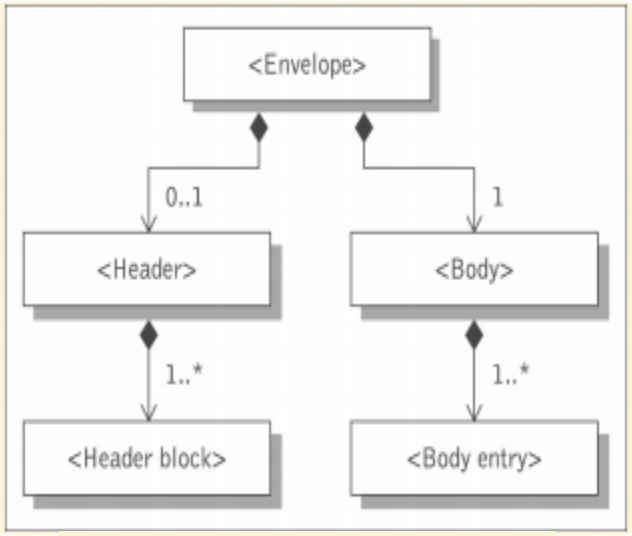
- SOAP permet donc l'échange d'information dans un environnement décentralisé et distribué, comme Internet par exemple.
- Il permet l'invocation de méthodes, de services, de composants et d'objets sur des serveurs distants
- Il peut fonctionner sur de nombreux protocoles (des systèmes de messagerie à l'utilisation de RPC).
- Il fonctionne particulièrement bien avec le protocole HTTP, protocole très souvent utilisé avec SOAP.

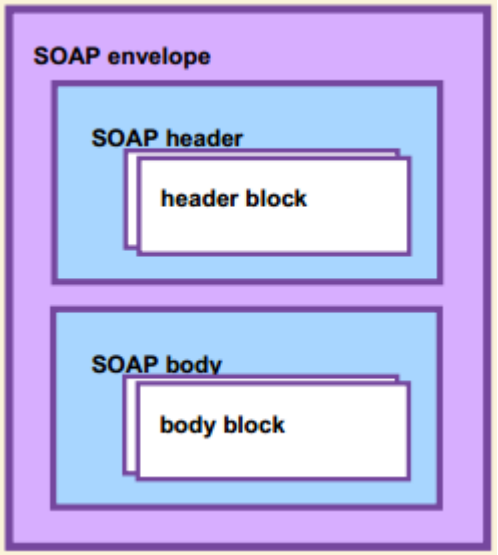
SOAP


Client
Procuration
D'interface
Message
SOAP
HTTP
Application cliente
Parser
XML
Serveur
APPEL
Translateur
SOAP
RPC
Local
Réponse
Parser
XML
Serveur d'applicaiton

SOAP – Message

- SOAP est basé sur l'échange de messages.
- Les messages sont considérés comme des enveloppes où les applications enferment les données à envoyer
- SOAP <Header>: il contient les bloques d'information concernant comment le message doit être traité. Ce qui contribue à transmettre l'information dans les messages SOAP qui ne concerne pas les application mais plutôt le moteur SOAP.
- SOAP <BODY>: l'information end-to-end principale est véhiculée dans le message SOAP;





UDDI

- Universal Description, Discovery and Integration
- Standard de l' « Organization for the Advancement oF Structured Information Standard » (avril 2003)
- UDDI permet aux fournisseurs de services de s'inscrire et de répertorier les services qu'ils proposent.
- UDDI ne contient que des références associées à des services, et non les services eux-mêmes.

UDDI

- Il peut être public ou privé
  - Les annuaires publics sont hébergés par des sociétés comme IBM ou Microsoft.
  - Les annuaires publics sont moins développés que les annuaires privés parce qu'ils ne sont pas suffisamment sécurisés.
  - Les annuaires privés peuvent être hébergés par une société quelconque sur un réseau privé ou sur internet

UDDI

- L'annuaire UDDI est composé de:
  - Pages blanches: Contiennent des informations sur l'entreprise comme le nom de la société, l'adresse
  - Pages jaunes: Contiennent la description des services web, au format WSDL, déployés par l'entreprise.
  - Pages vertes : Contiennent les informations techniques détaillées sur les services fournis (processus métier, description de service….)

UDDI

- Pages blanches ou businessEntity:
  - Décrites sous la forme d'un schéma XML.
  - Elles contiennent les éléments relatifs à l'entreprise qui propose le service (nom, coordonnées, secteur d'activité, l'adresse du site web…)

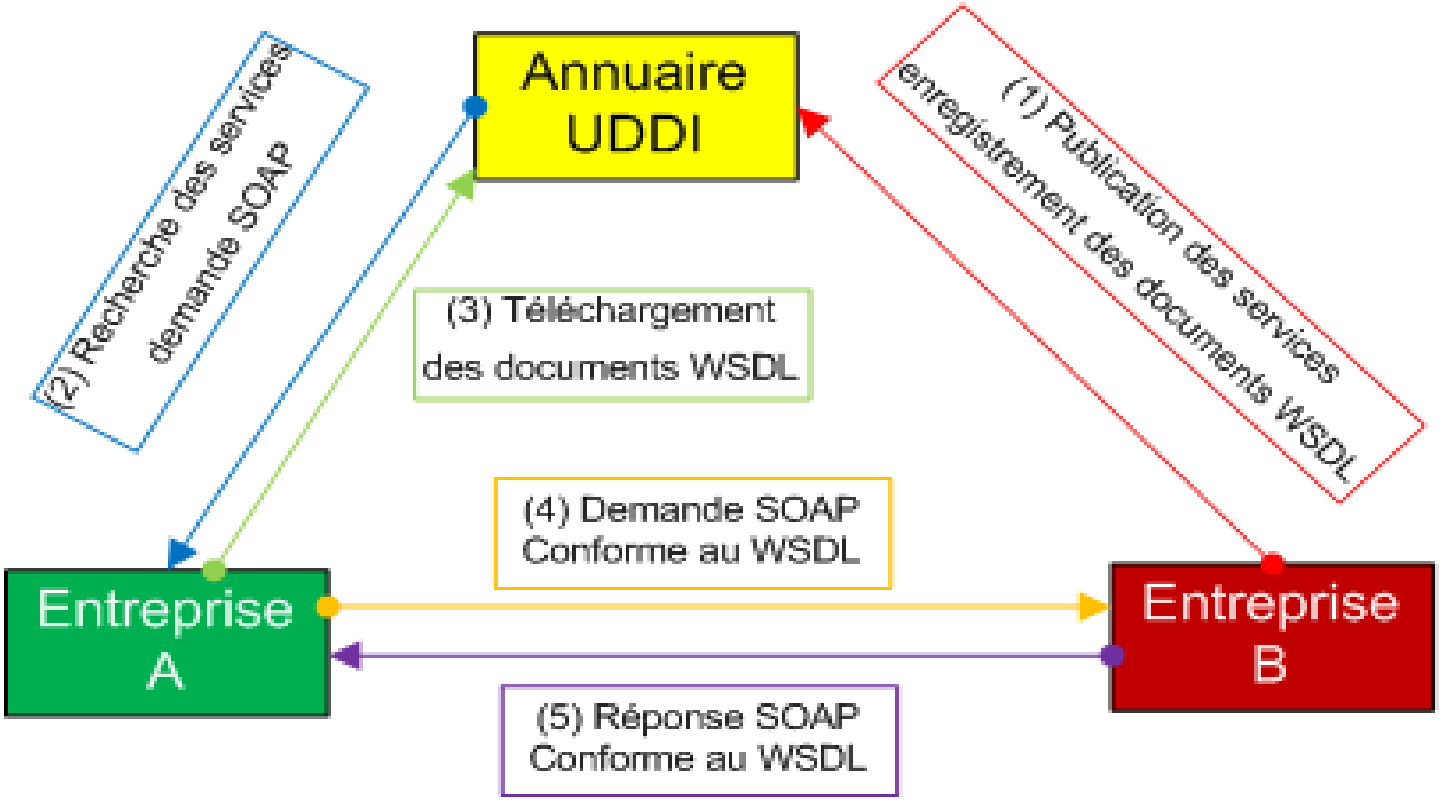
UDDI

- Pages jaunes ou ServiceEntity:
  - Décrites aussi sous la forme d'un schéma XML
  - C'est un ensemble de services proposés répondant à un besoin métier spécifique
  - Contiennent aussi la description des services web proposés par ce dernier (nom du service, description, code…)
  - Une entreprise peut avoir plusieurs métiers et donc plusieurs businessService.

UDDI

pages vertes:

- Contiennent les informations techniques sur un service web.
- Contiennent aussi les références aux tmodels (spécification des interfaces des services web)


Annuaire
UDDI
(1) Publication des services
enregistrement des documents WSDL
(2) Recherche des services
demande SOAP
(3) Téléchargement
des documents WSDL
(4) Demande SOAP
Conforme au WSDL
Entreprise
A
Entreprise
B
(5) Réponse SOAP
Conforme au WSDL

Les web services

- Les Web Services sont des services offerts via le web.

- Les **services web** représentent un mécanisme de communication entre applications distantes à travers le réseau internet indépendant de tout langage de programmation et de toute plate-forme d'exécution.

Les web services

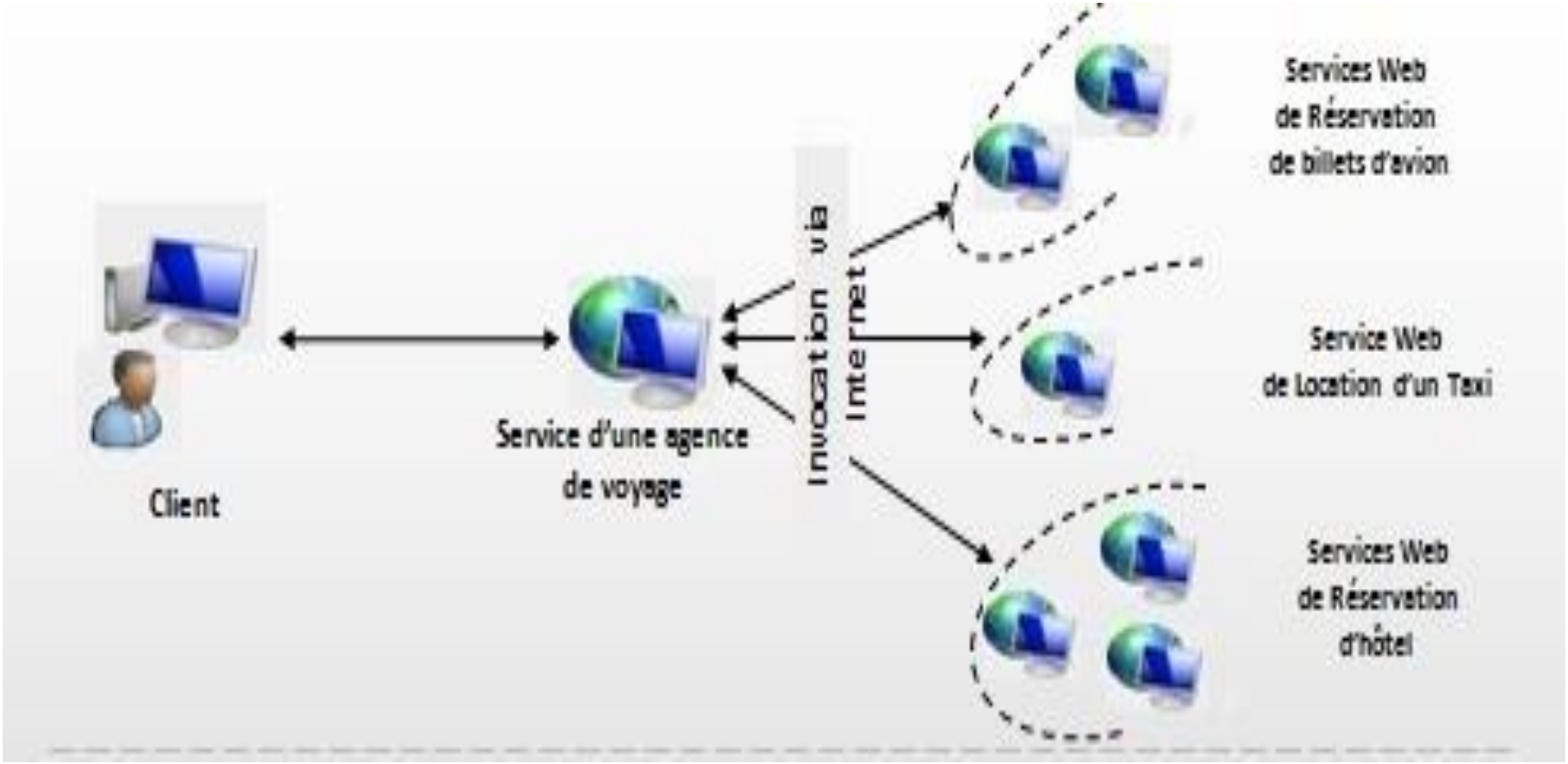
- utilisant le protocole HTTP comme moyen de transport.

- Ainsi, les communications s'effectuent sur un support universel, maîtrisé et généralement non filtré par les pare-feux ;employant une syntaxe basée sur la notation XML pour décrire les appels de fonctions distantes et les données échangées ; organisant les mécanismes d'appel et de réponse.

Les web services

- Les services web facilitent non seulement les échanges entre les applications de l'entreprise mais surtout permettent une ouverture vers les autres entreprises.
- Les premiers fournisseurs de services web sont ainsi les fournisseurs de services en ligne (météo, bourse, planification d'itinéraire, pages jaunes, etc.), mettant à disposition des développeurs des API (Application Programmable Interface) payantes ou non, permettant d'intégrer leur service au sein d'applications tierces.

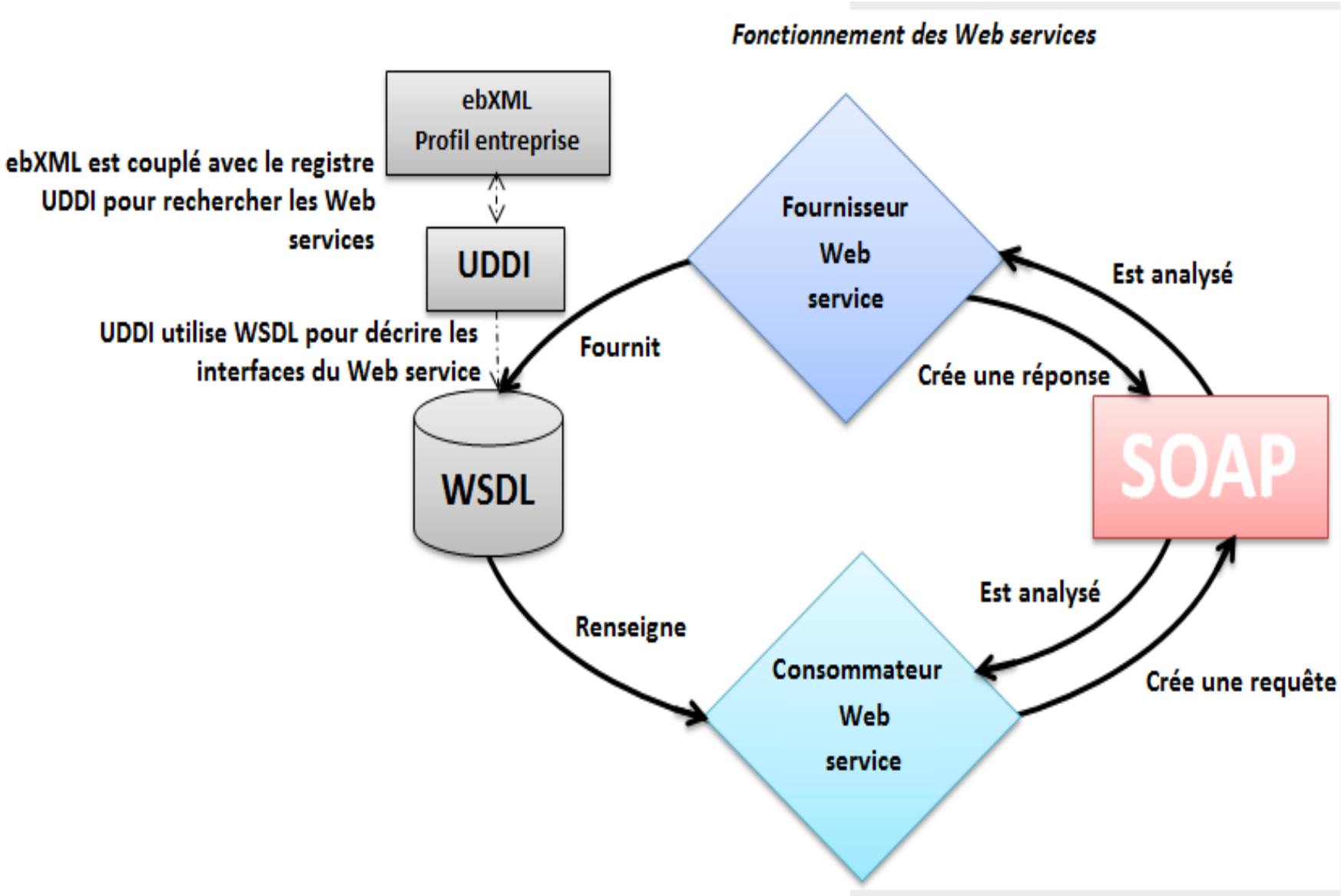
Web service- Exemple



Les couches de Web Service

- Le fonctionnement des services web repose sur un modèle en couches, dont les trois couches fondamentales sont les suivantes :
- **Invocation**, visant à décrire la structure des messages échangés par les applications.
- **Découverte**, pour permettre de rechercher et de localiser un service web particulier dans un annuaire de services décrivant le nom de la société, l'objectif de chaque service, etc.
- **Description**, dont l'objectif est la description des interfaces (paramètres des fonctions, types de données) des services web

Fonctionnement des web services



EbXML: Electronic Business using eXtensible Markup Language **signifiant** ***commerce électronique en utilisant XML*****, est une suite de spécifications basées sur le langage XML utilisable pour le commerce électronique.**

L'objectif est de fournir une infrastructure globale, ouverte, fondée sur XML, permettant d'assurer les échanges électroniques professionnels (B2B) de manière interopérable.

SOA et web services

- Attention à ne pas confondre les 2 !
  - SOA est un ensemble de concepts :
    Une SOA peut se mettre en œuvre sans Web Services
  - Les WS sont de l'ordre de la technologie :
    On peut utiliser les Web Services sans faire de SOA
- Les WS constituent la meilleure solution standardisée disponible
  - Un service métier = un webservice

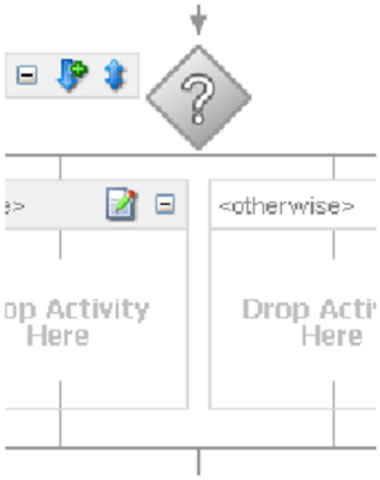
Le langage BPEL

- Standard de l'OASIS : (*Organization for the Advancement of Structured Information Standards*: est un consortium mondial qui travaille pour la standardisation de formats de fichiers ouverts basés notamment sur XML)
- Langage graphique
- Norme permettant de décrire des processus en XML Propose les fonctions basiques d'un langage de programmation:
  - sequence, flow, loop, switch…
- Identification des Instances de Process
- Gestion des fautes

Activités BPEL de base (1/3)

Nom	Symbole	Description
Assign		Permet de manipuler les variables d'un processus ➔Initialisation de variable ➔Copie de variable ➔Manipulation XML (XPath)
Scope		Découpe de manière logique un processus BPEL. Il permet : ➔De déclarer des variables locales ➔De récupérer des exceptions
Flow		Exécute différents traitements en parallèle.

Activités BPEL de base (2/3)

Nom	Symbole	Description
Switch		➔Fournit n branches. ➔Chacune de ces branches est associée à une expression booléenne. ➔La première branche (de gauche à droite) dont la condition est vraie est exécutée. ➔Il est possible de fournir une branche otherwise qui est lancée si aucune des autres branches n'est valide.
While		Effectue un traitement tant que la condition associée est respectée.

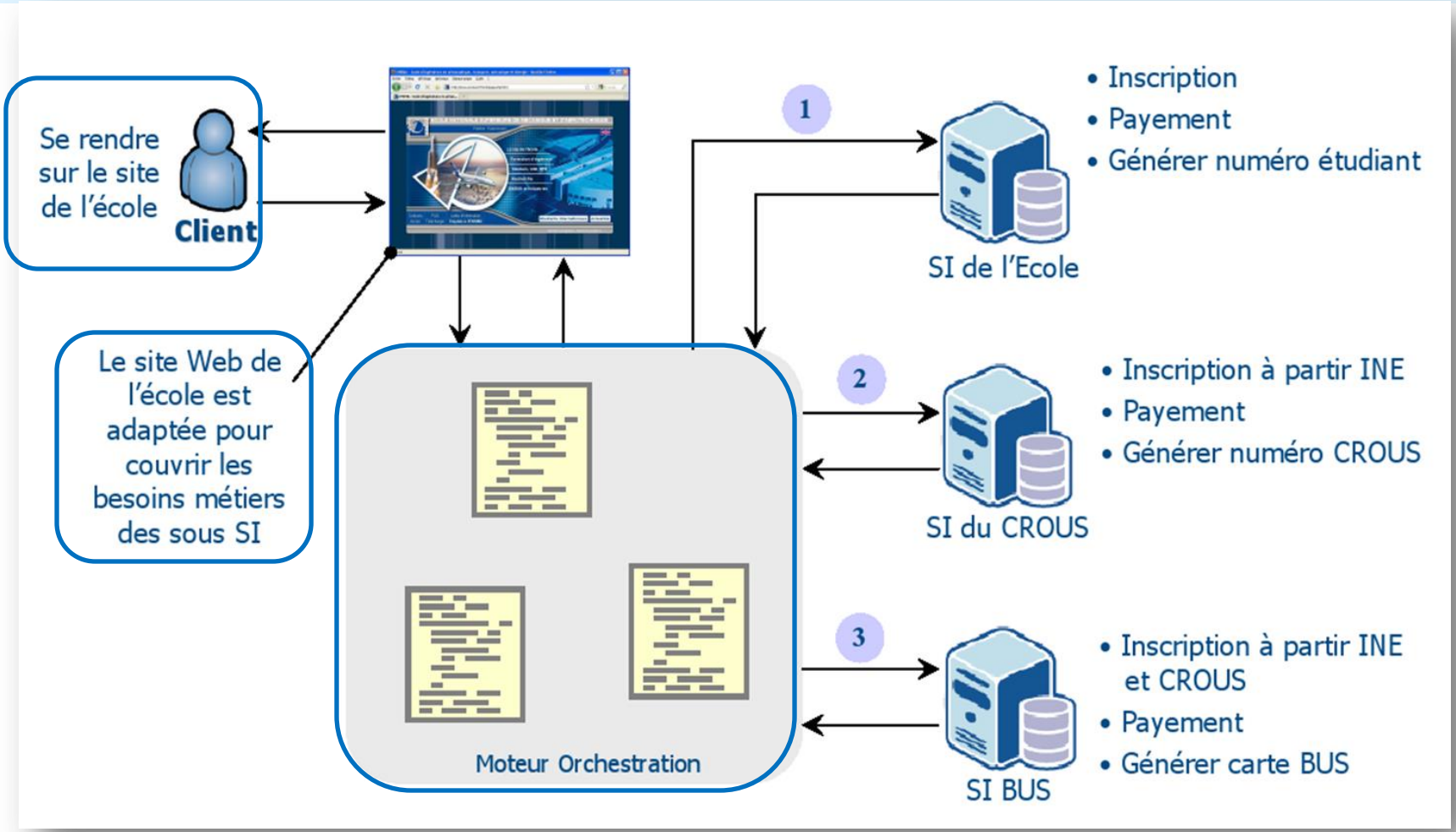
Activités BPEL d'orchestration

Nom	Symbole	Description
Invoke		Appelle un web service partenaire (PartnerLink)
Receive		Permet de recevoir un message d'un web service partenaire
Reply		Envoie un message (ou une exception) en sortie de processus.
Pick		Permet de combiner plusieurs Receive et Timeout (Une seule branche peut être prise en compte).

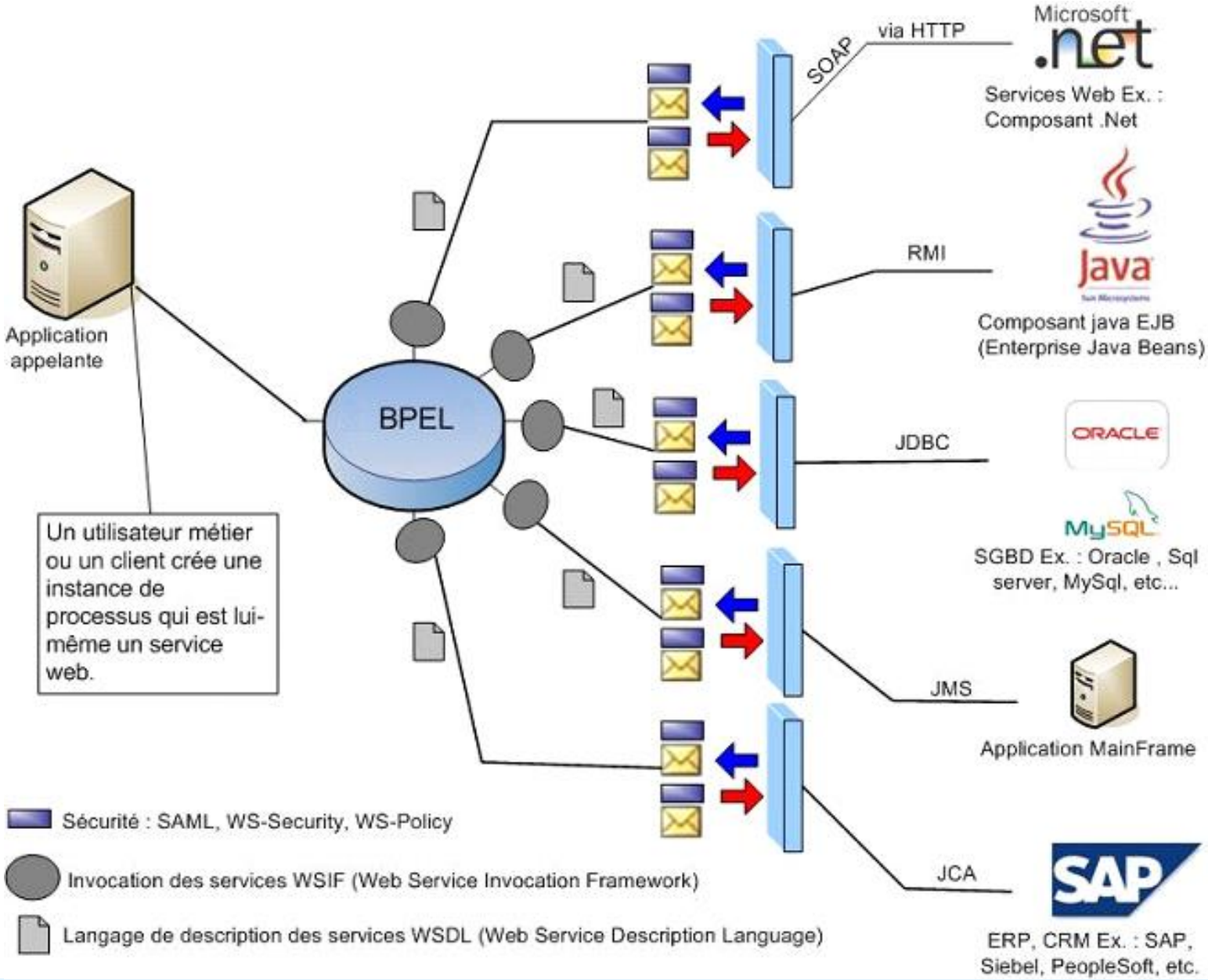
BPEL-Partner Links

- Du point de vue des clients le processus BPEL est un Service Web
- Deux façons d'interagir entre un processus BPEL et des Web Services externes
  - Un processus BPEL invoque des opérations issues d'autres Web Services (dit Partenaires) : Lien de partenaire (PartenerLink) de type invocation
  - Un processus BPEL reçoit des invocations issues de clients : Lien de partenaire (PartenerLink) de type client
- Pour résumer, un lien de partenaire désigne les relations entre des partenaires / clients et le processus BPEL

BPEL


Se rendre sur le site de l'école
Client
Le site Web de l'école est adaptée pour couvrir les besoins métiers des sous SI
Moteur Orchestration
1
2
3
SI de l'Ecole
• Inscription
• Payement
• Générer numéro étudiant
SI du CROUS
• Inscription à partir INE
• Payement
• Générer numéro CROUS
SI BUS
• Inscription à partir INE et CROUS
• Payement
• Générer carte BUS

BPEL le chef d'orchestre



Bon courage